Three instructions

Digital Systems – Lecture 23

Department of
COMPUTER
SCIENCE

In this lecture

Using the datapath to implement three instructions.

- `adds r1, r2, r3`
- `str r0, [sp, #48]`
- `bgt .-4`

An ALU operation

```
adds r1, r2, r3
```

An ALU operation

between registers

function is +

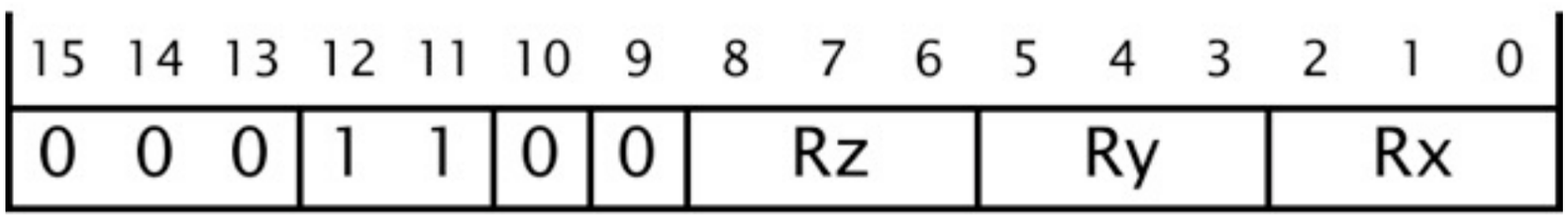
```
00011 00 011 010 001
```

source 2 is r3

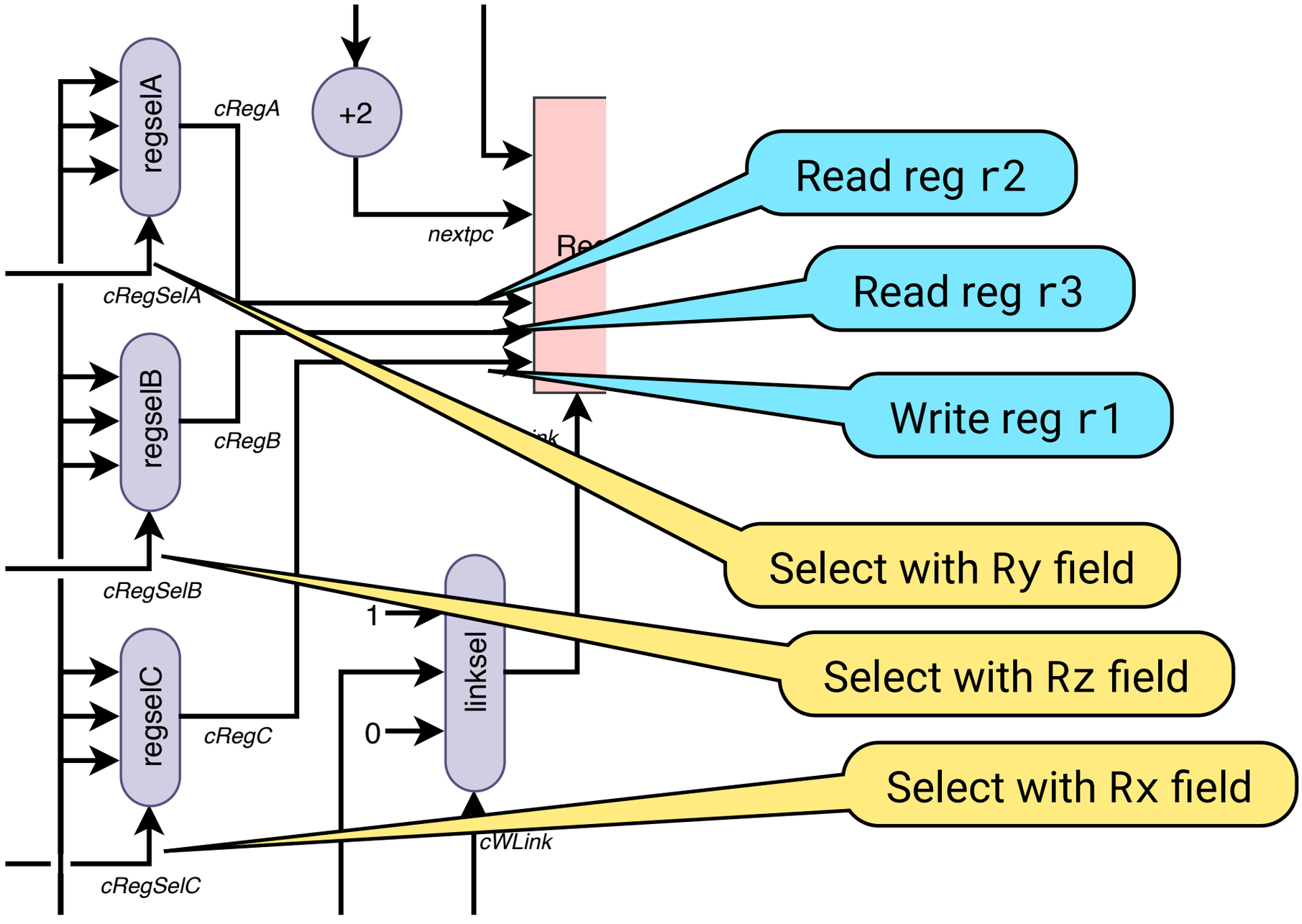
source 1 is r2

destination is r1

adds ⟨Rx⟩,⟨Ry⟩,⟨Rz⟩



Selecting registers

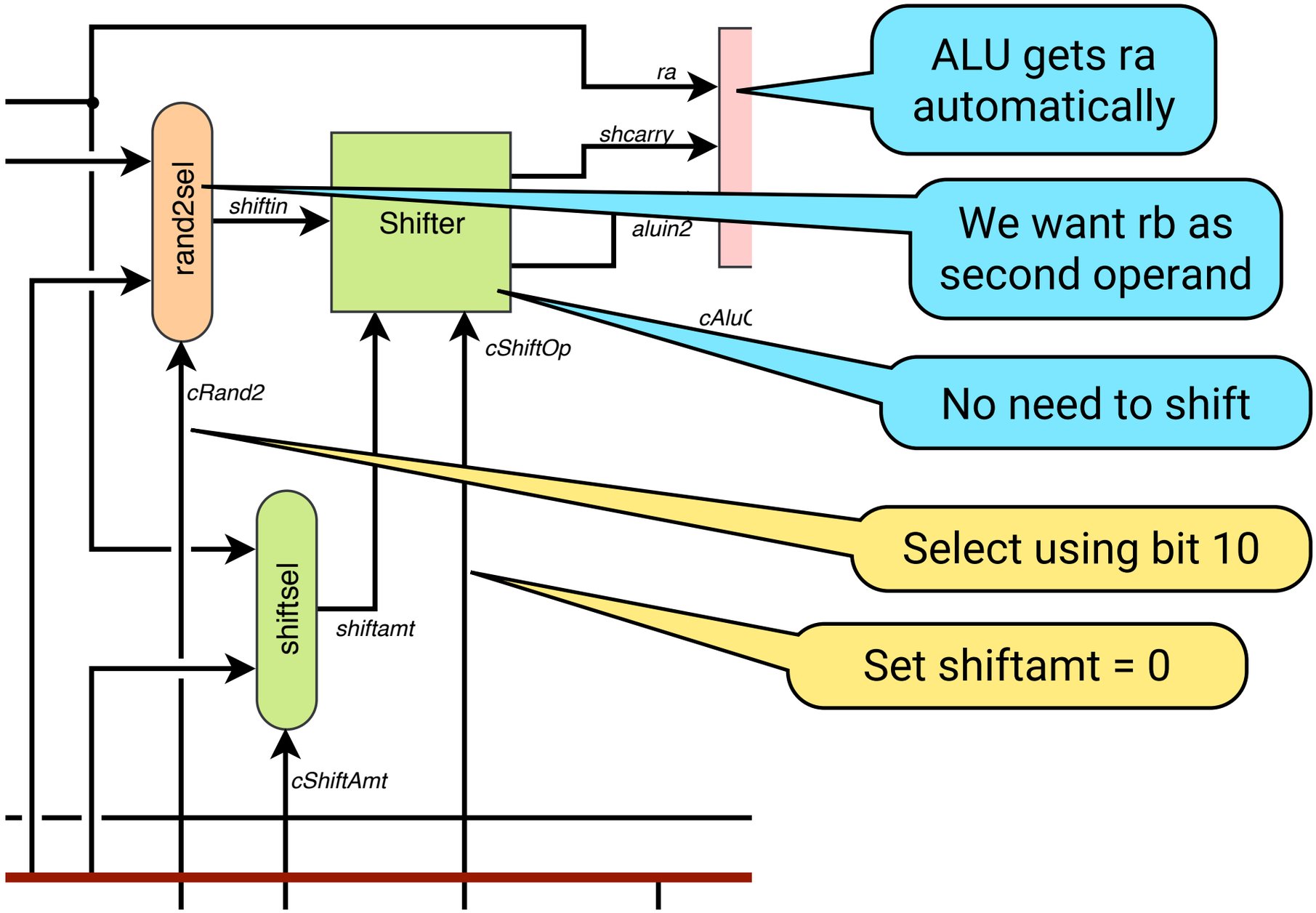


Rule 3 of 0 … 31

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
3 :	adds/subs	**Ry/Rz/Rx**					

- Use field Ry[3:5] to select the first register to be read.
- Use field Rz[6:8] to select the second register to be read.
- Use field Rx[0:2] to select the register to write.

Feeding the ALU

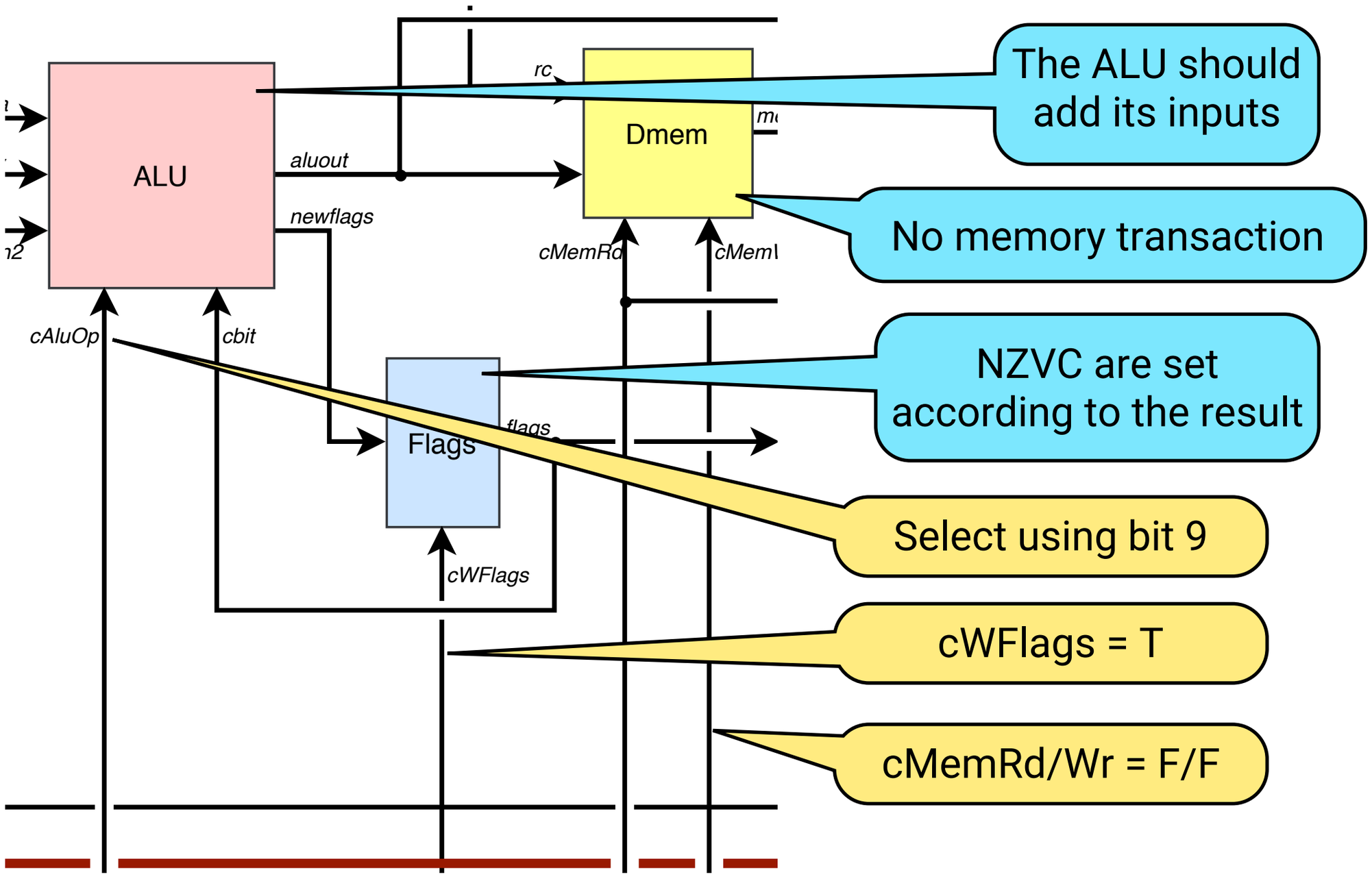


In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
3:	adds/subs	Ry/Rx/Rz	**RImm3**	**Lsl/Sh0**			

- Unusually, use bit 10 of the instruction to choose between `rb` and a 3-bit immediate.
- Shift the second operand (left) by 0 bits.

Performing the operation

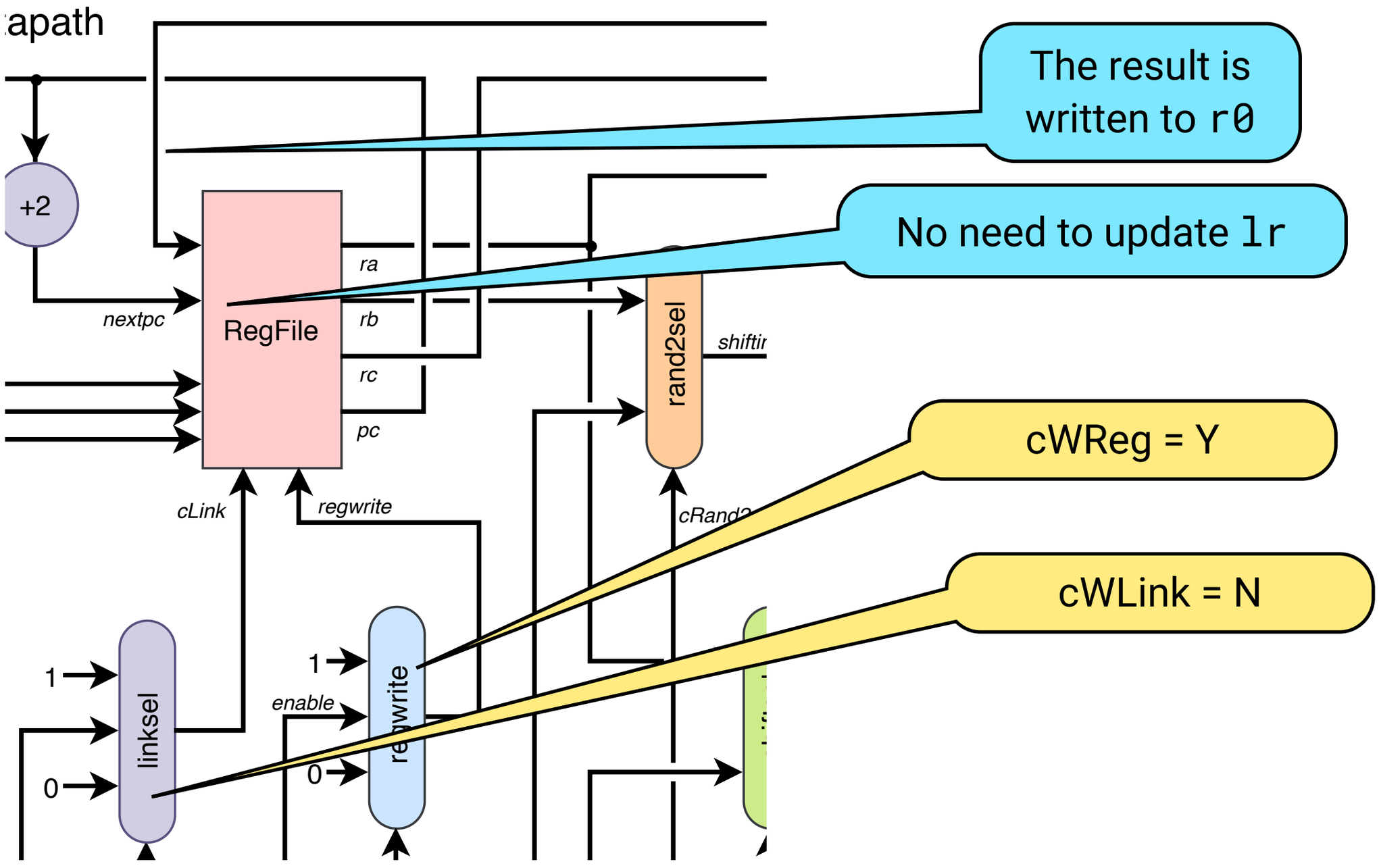


In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
3:	adds/subs	Ry/Rx/Rz	RImm3	Lsl/Sh0	**Bit9**	**F/F**	**T/ /**

- Unusually, use bit 10 of the instruction to choose between `rb` and a 3-bit immediate field.
- Shift the second operand (left) by 0 bits.

Writing the result



In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
3:	`adds/subs`	`Ry/Rx/Rz`	`RImm3`	`Lsl/Sh0`	`Bit9`	`F/F`	`T/`**`Y`**`/`**`N`**

- Write the result (from the ALU) into the third selected register.
- But don’t write `lr` with the address of the next instruction.

A store instruction

```
str r0, [sp, #48]
```

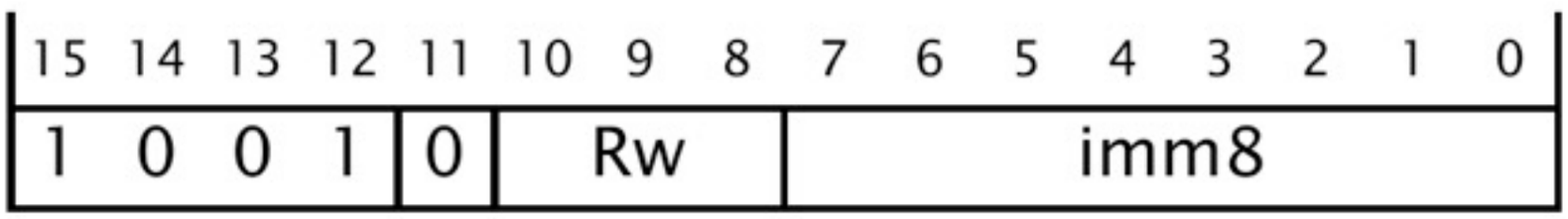
A special form of store

data from r0

```
10010 000 00001100
```

offset 4*12 from sp

str ⟨Rw⟩,[sp,#⟨imm8⟩]

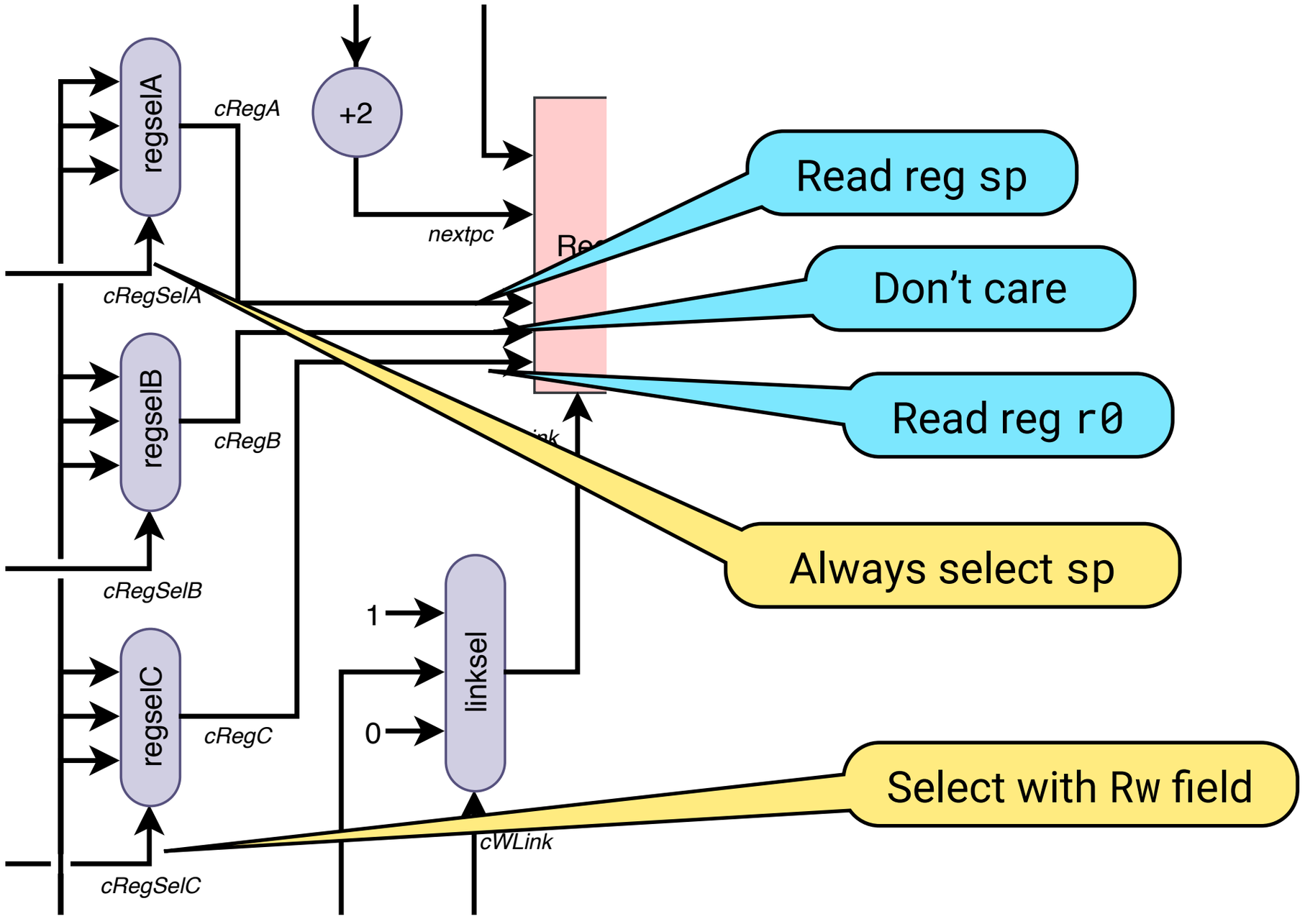


The plan

`str r0, [sp, #48]`

- Read registers `sp` and `r0`.
- Use the shifter and ALU to compute the effective address.
- Perform a write cycle with the data memory.

Selecting registers

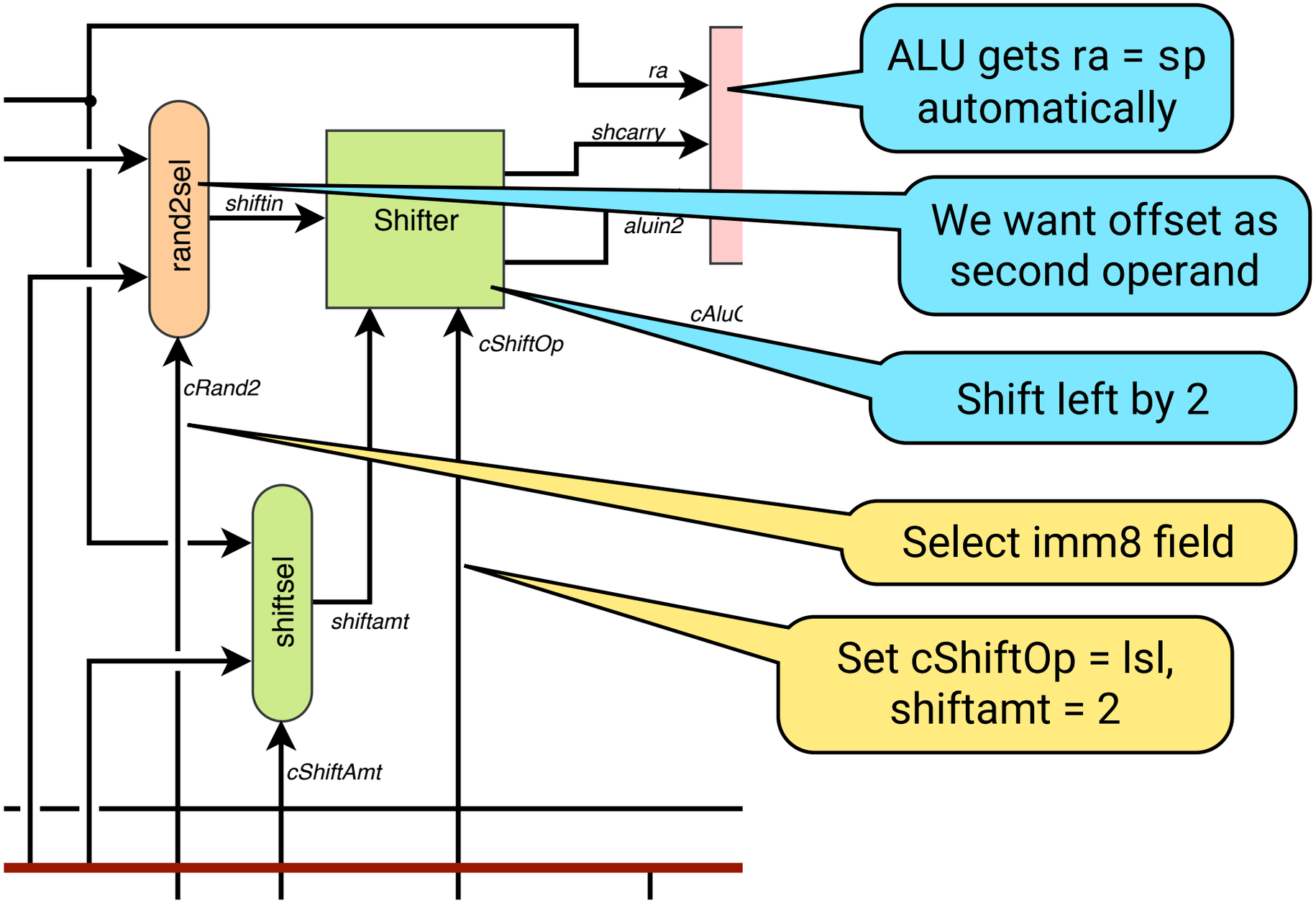


Rule 18 of 0 … 31

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18:	`str sp`	**Rsp/_/Rw**					

- Always select sp as the first register to be read.
- Some register will be the second register, but we don’t care which.
- Use field Rw[8:10] to select the third register to read.

Feeding the ALU

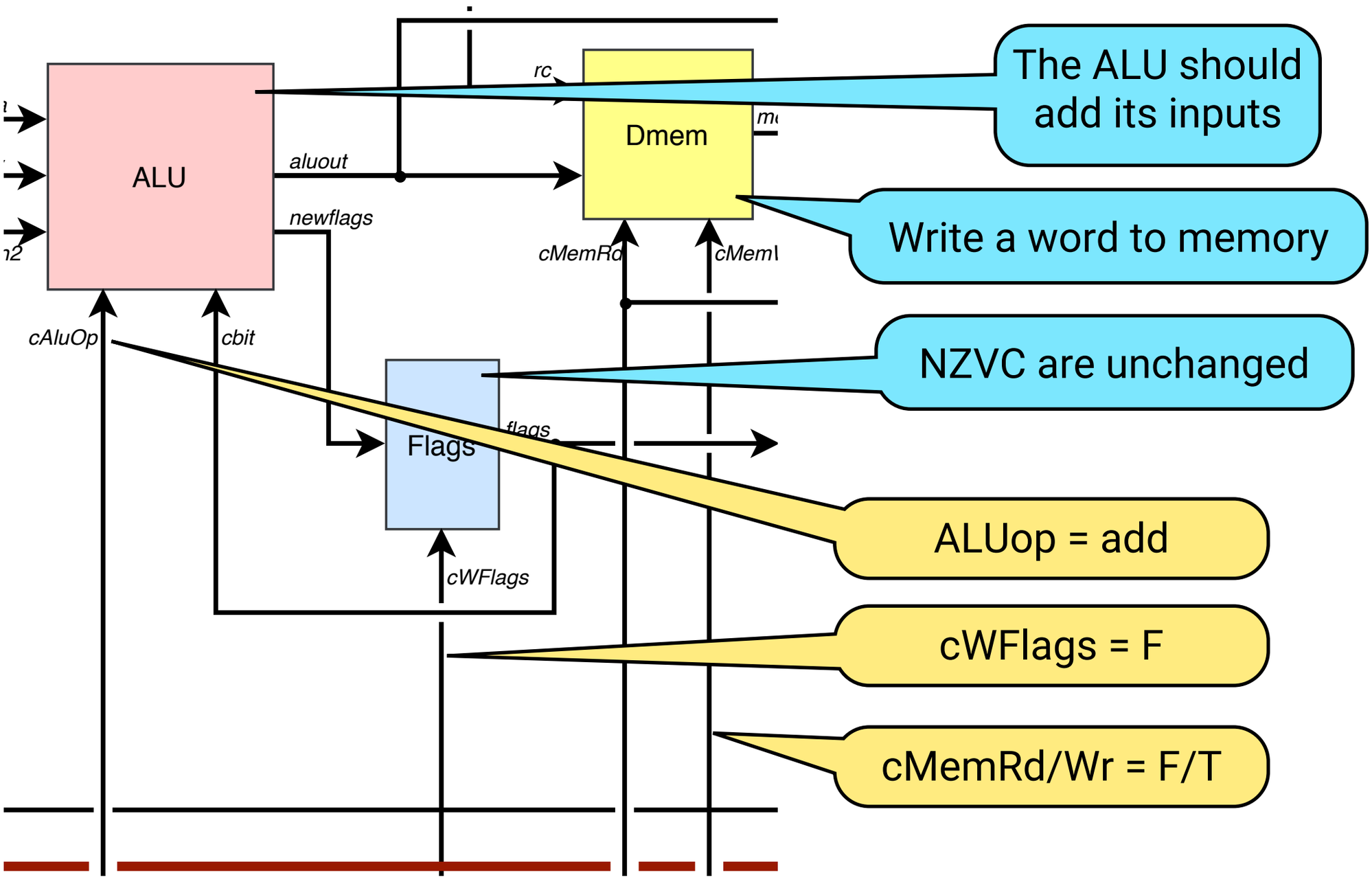


In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18:	`str sp`	`Rsp/_/Rw`	**`Imm8`**	**`Lsl/Sh2`**			

- The second ALU input comes from an immediate field.
- The value is scaled by 4, shifting left a constant 2 places.

Performing the operation



In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18:	`str sp`	`Rsp/_/Rw`	`Imm8`	`Lsl/Sh2`	**Add**	**F/T**	**F/N/N**

- The ALU adds sp and scaled offset to form the effective address.
- The value from the third register is written to that memory word.
- The flags are unchanged, no result is written to a register, and `lr` is not written.

A conditonal branch

```
bgt  .-4
```

Conditonal branch

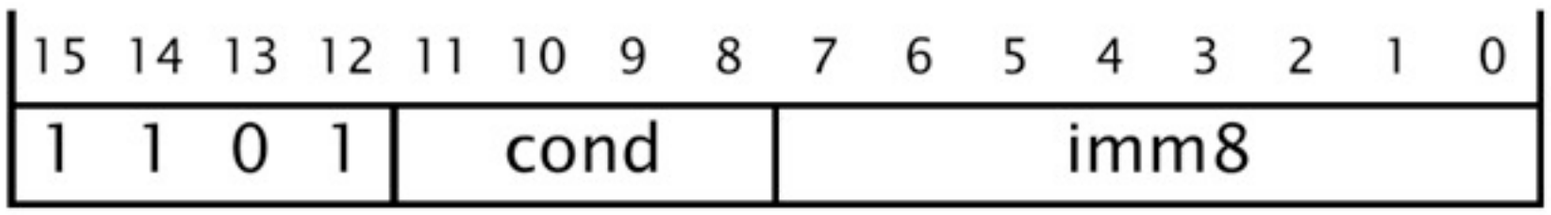
if greater than

```
label:
   sub r0, r0, #1
   cmp r0, #100
   bgt label
```

```
1101 1100 11111100
```

offset 2*(−4)
relative to pc+4

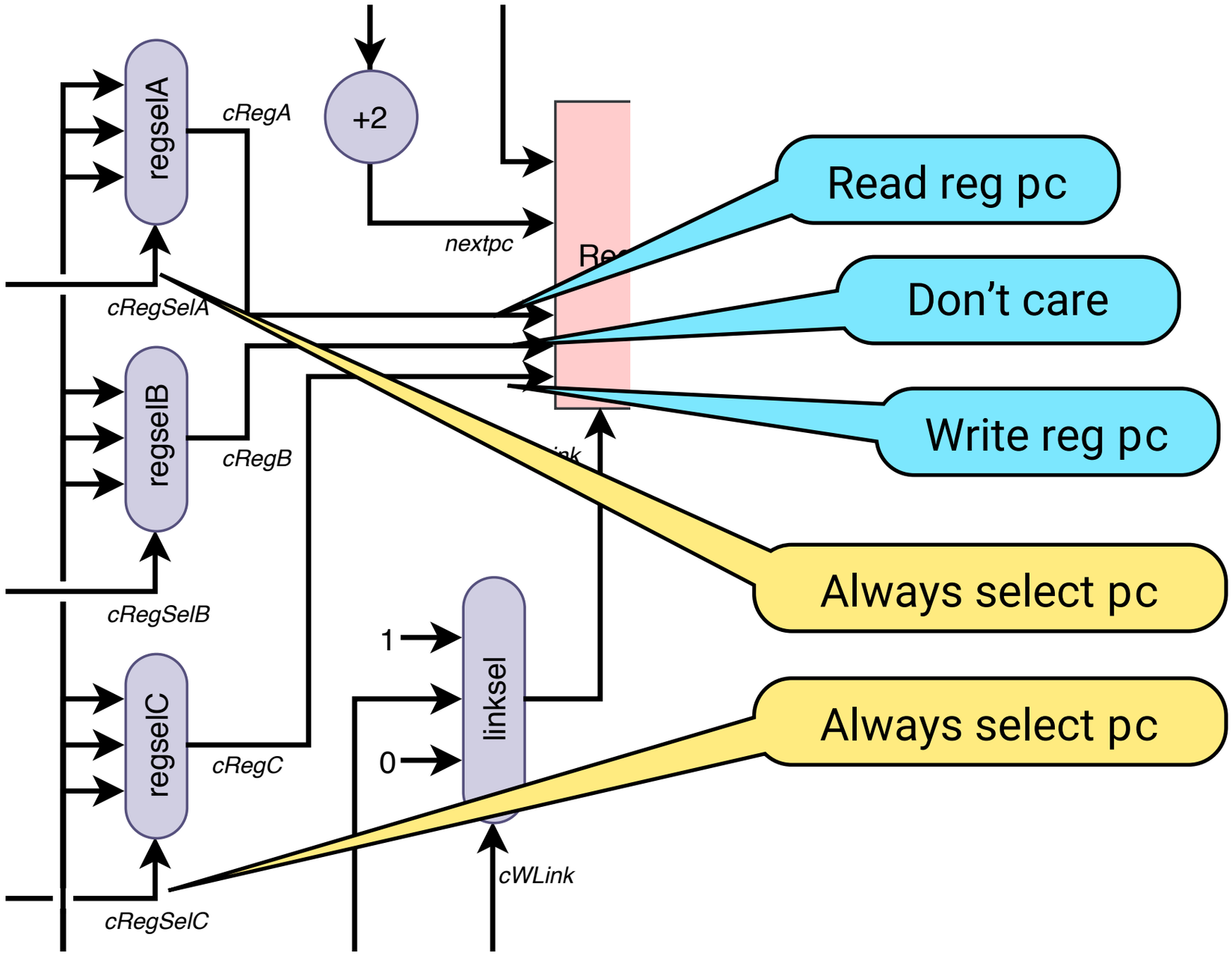
b⟨c⟩ ⟨imm8⟩



The plan

- Use the ALU to compute the branch target address, whether needed or not.
- Use the conditional execution unit to write back the result into the pc only of the branch is taken.

Selecting registers

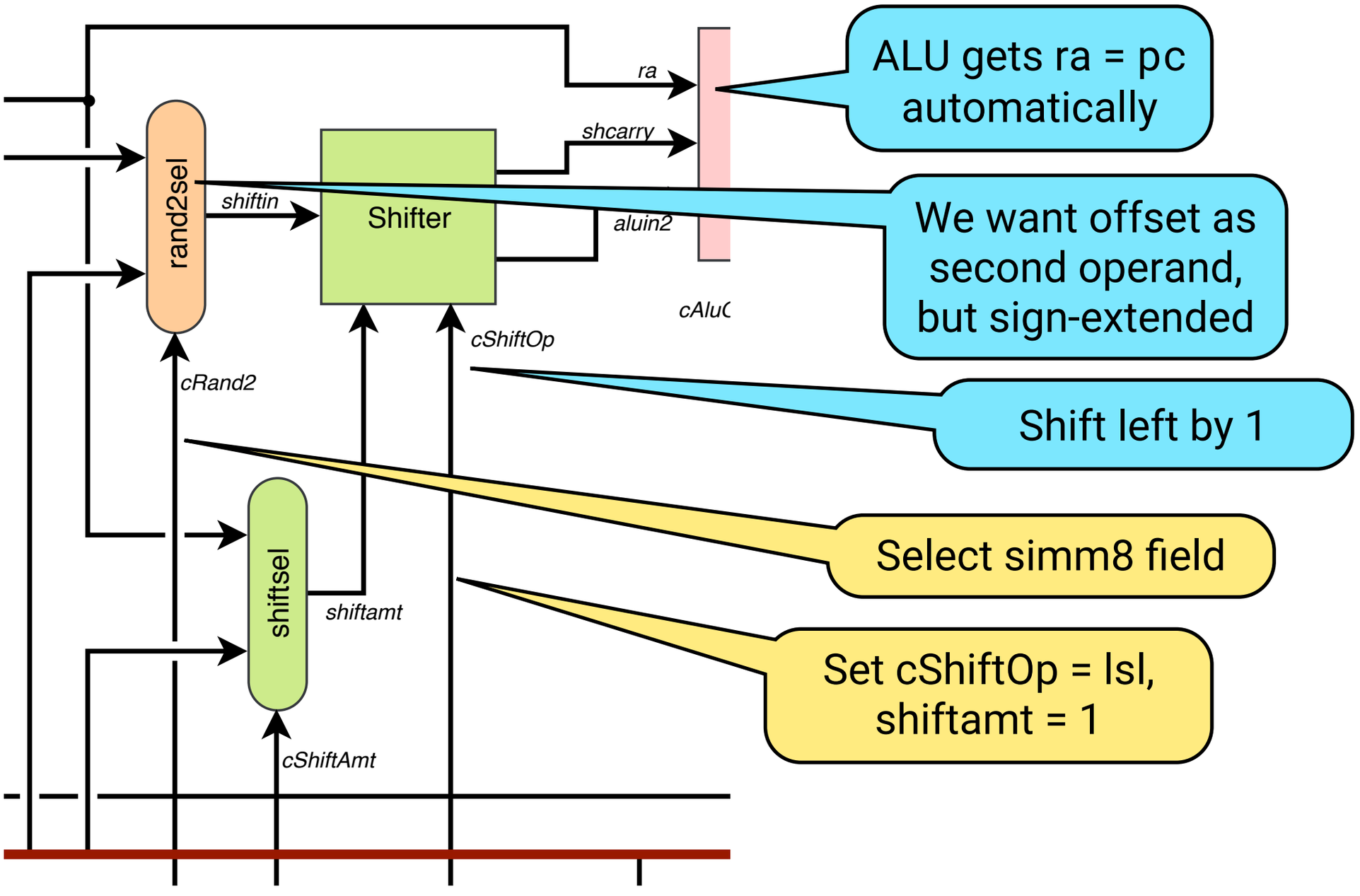


Rules 26–27 of 0 … 31

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18 : 19 :	b<cond>	**Rpc / _ / Rpc**					

- Always select pc as the first register to be read, and the register to be (conditionally) written.

Feeding the ALU

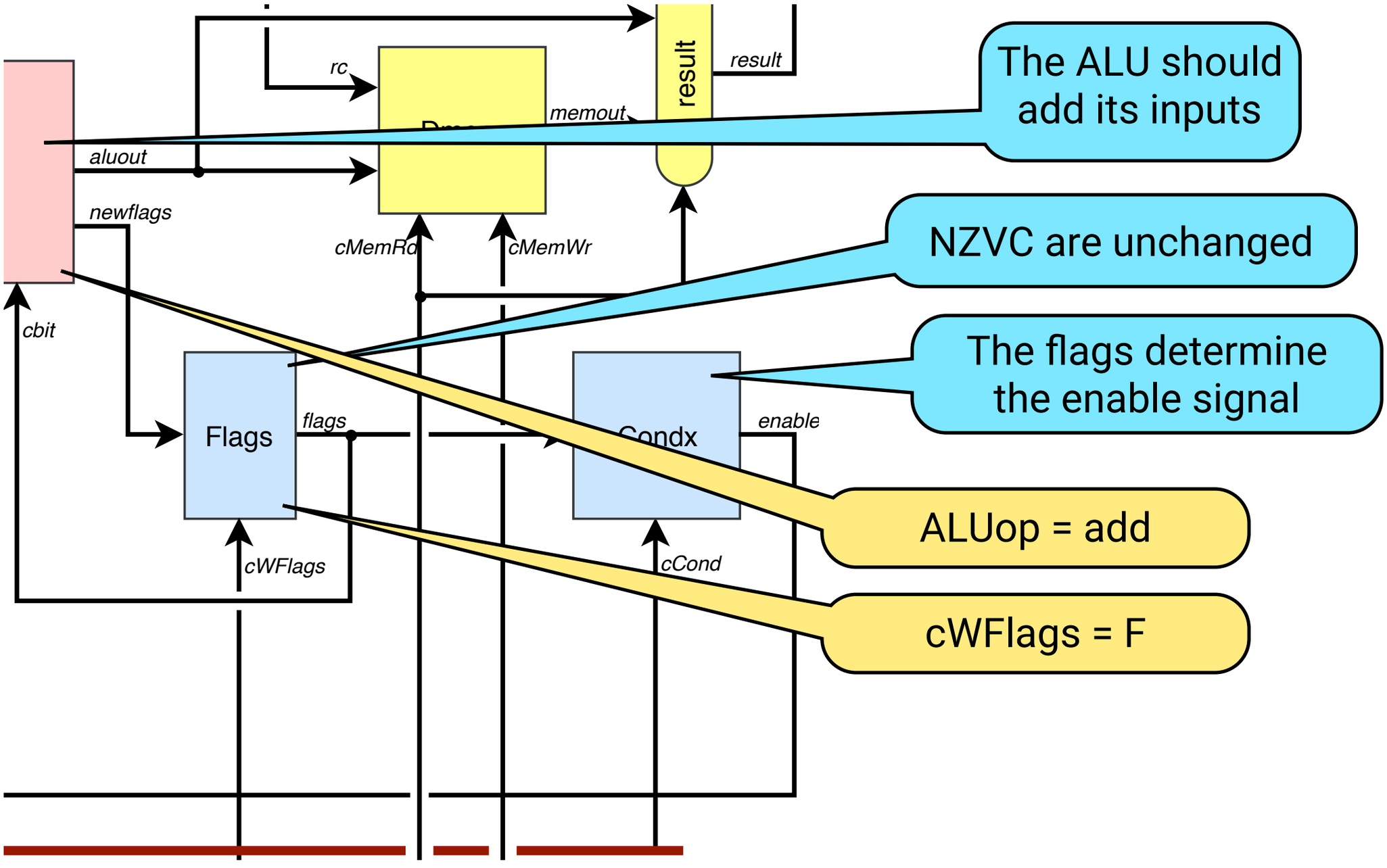


In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18: 19:	b<cond>	Rpc/_/Rpc	**Simm8**	**Lsl/Sh1**			

- Select immediate field as second operand, with sign-extension.
- Scale by 2, shifting left by 1 place.

Performing the operation

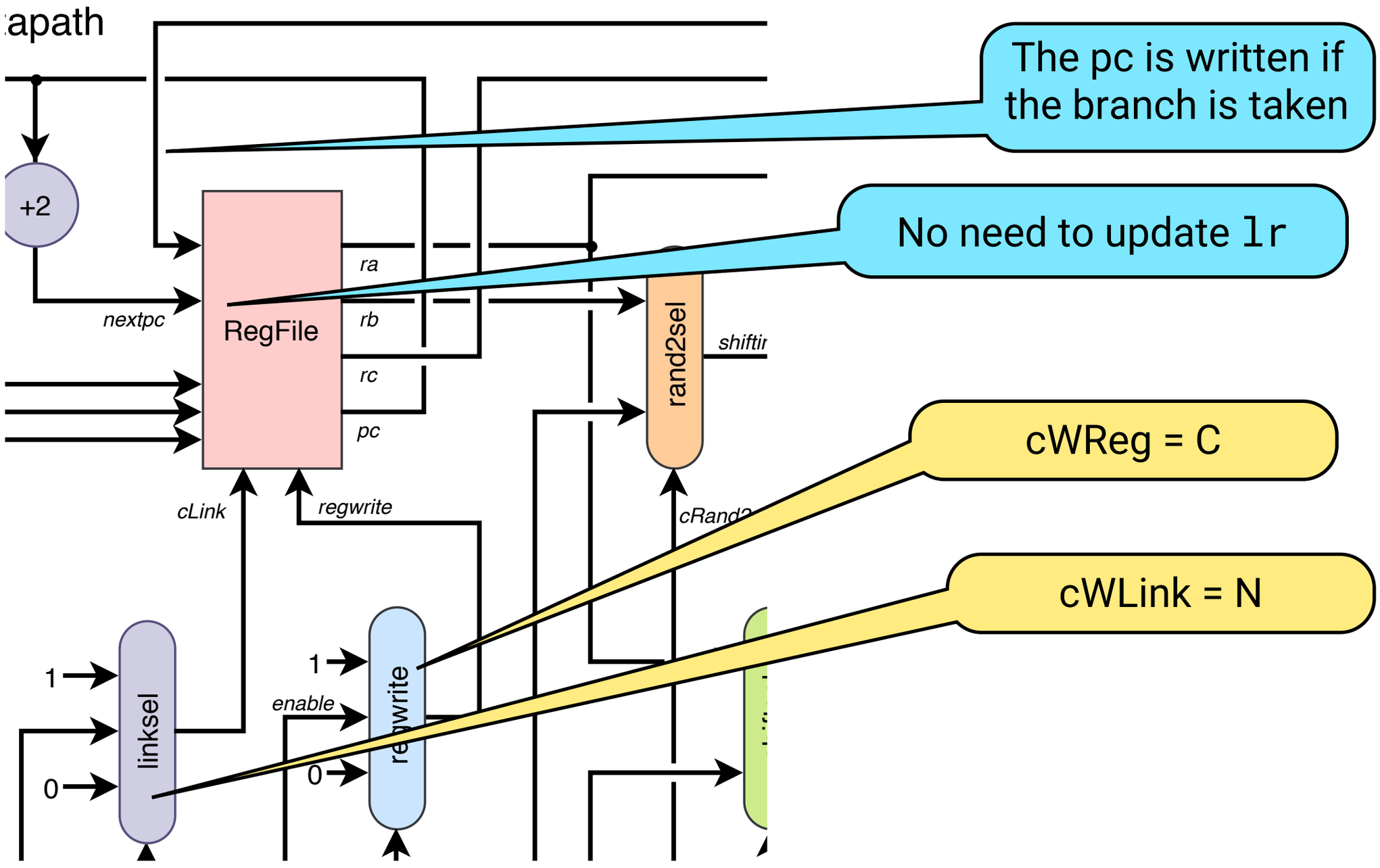


In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18: 19:	b<cond>	Rpc/_/Rpc	Simm8	Lsl/Sh1	**Add**	**F/F**	**F/ /**

- The ALU adds pc+4 and the scaled and sign-extended offset.
- No memory transaction needed.
- The flags are used but not changed.

Conditional execution



In the control word

	Instruction	cRegSel A/B/C	cRand2	cShiftOp/ Amt	cAlu Sel	cMem Rd/Wr	cWFlags/ Reg/Link
18: 19:	b<cond>	Rpc/_/Rpc	Simm8	Lsl/Sh1	Add	F/F	F/**C**/**N**

- The branch target is always computed.
- It is written into the `pc` if the enable signal is true.
- There's no need to update the `lr` register.